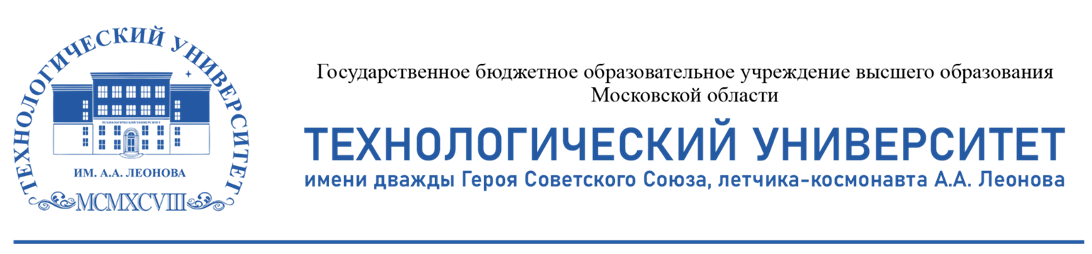
**Колледж космического машиностроения и технологии**



**ОТЧЕТ**

по производственной практике ПП.03.01

по профессиональному модулю ПМ.03 «Участие в интеграции программных модулей»

специальность 09.02.03 «Программирование в компьютерных системах»

Выполнил студент

Ларченко Михаил Андреевич

группы: П1-18

Проверил преподаватель

Бобкова Н. Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2021

**Содержание**

[Введение 3](#_Toc93676135)

[1. Предоставить общие сведения о предприятии. 4](#_Toc93676136)

[1.1. Сферу деятельности предприятия, история появления и развития предприятия. 4](#_Toc93676137)

[1.2. Структуру предприятия, функции подразделения и взаимосвязь с другими подразделениями. 5](#_Toc93676138)

[1.3. Список задач подразделения в сфере информационных технологий. 5](#_Toc93676139)

[2. Содержание выполняемых видов работ. 5](#_Toc93676140)

[2.1. Анализ существующих в подразделении локальных и распределенных баз данных, приложений баз данных 5](#_Toc93676141)

[2.2. Получение задания на конкретную разработку 5](#_Toc93676142)

[2.3. Изучение документооборота, подлежащего автоматизации 6](#_Toc93676143)

[2.4 Построение IDEF-модели 6](#_Toc93676144)

[2.5.1. Наименование программы 7](#_Toc93676145)

[2.5.2. Краткая характеристика области применения 7](#_Toc93676146)

[2.5.3. Основания для разработки 7](#_Toc93676147)

[2.5.4. Функциональное назначение 7](#_Toc93676148)

[2.5.5. Эксплуатационное назначение 7](#_Toc93676149)

[2.5.7. Требования к организации входных и выходных данных 8](#_Toc93676150)

[2.5.8. Требования к временным характеристикам 8](#_Toc93676151)

[2.5.9. Требования к обеспечению надежного (устойчивого) функционирования программы 8](#_Toc93676152)

[2.5.10. Время восстановления после отказа 8](#_Toc93676153)

[2.5.11. Отказы из-за некорректных действий оператора 8](#_Toc93676154)

[2.5.12. Условия эксплуатации 8](#_Toc93676155)

[2.5.12. Требования к составу и параметрам технических средств. 8](#_Toc93676156)

[2.5.13. Технико-экономические показатели 9](#_Toc93676157)

[2.5.14. Стадии и этапы разработки 9](#_Toc93676158)

[2.5.15. Порядок контроля и приемки 9](#_Toc93676159)

[2.6. Разработать модули программного средства в соответствии с ТЗ. 9](#_Toc93676160)

[2.6.1. Описание главного модуля. 9](#_Toc93676161)

[2.6.2. Описание спецификаций к модулям 10](#_Toc93676162)

[2.6.3. Описание модулей 10](#_Toc93676163)

[2.6.4. Описание тестовых наборов модулей 13](#_Toc93676164)

[2.7. Разработать пользовательский интерфейс программного средства. 13](#_Toc93676165)

[2.8. Провести отладку и тестирование программного средства. 16](#_Toc93676166)

[2.9. Разработать комплект технологической документации. 17](#_Toc93676167)

[3. Проектирование 17](#_Toc93676168)

[Заключение 18](#_Toc93676169)

[Дневник производственной практики на предприятии «ККМТ» 18](#_Toc93676170)

[Дневник производственной практики на предприятии 18](#_Toc93676171)

[Приложение 1. Руководство оператора 18](#_Toc93676172)

[Приложение 2. Руководство программиста 18](#_Toc93676173)

[Приложение 3. Листинг кода 18](#_Toc93676174)

# Введение

При обучении на 4 курсе МГОТУ “Колледж космического машиностроения и технологий”, студентами группы П1-18 Волковым Максимом, Денисовым Матвеем и Курбанмухаммедовым Стивом, Ларченко Михаилом была пройдена практика по модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем».

**Задание на производственную практику**

# 1. Предоставить общие сведения о предприятии.

## 1.1. Сферу деятельности предприятия, история появления и развития предприятия.

Детские технопарки «Кванториум» — это федеральная сеть образовательных площадок, оснащенных высокотехнологичным оборудованием, где дети учатся по принципу проектного обучения: от теории сразу к практике.

Детский технопарк «Кванториум» — уникальная среда для ускоренного развития ребенка по актуальным научно-исследовательским и инженерно-техническим направлениям, оснащенная высокотехнологичным оборудованием. Мы обучаем детей универсальным навыкам и предметным компетенциям через решение реальных кейсов от промышленных партнеров по ключевым проблемам развития науки и техники и привитие 4K-компетенций (креативности, коммуникативности, критического мышления, умения работать в команде).

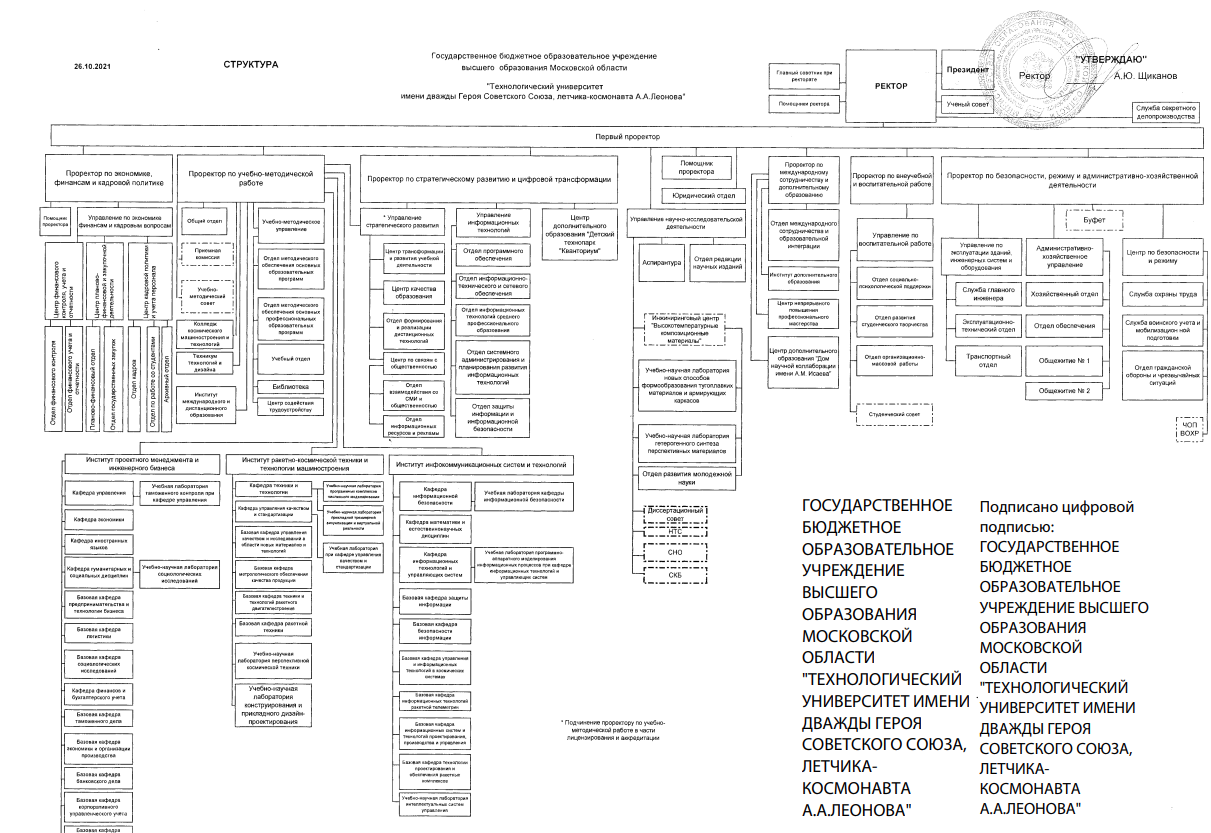
Кванториум — это школа мышления, основная цель которой — воспитать поколение специалистов, способных обеспечить технологический прорыв в нашей стране. Мы учим детей учиться: наставник не передает ребенку накопленные знания, а учит его задавать вопросы, работать с большим объемом информации и верифицировать ее.

Открытие первого в Московской области детского технопарка «Кванториум» состоялось на базе ГБОУ ВО МО «Технологический университет» в ноябре 2016 года. Под руководством высококвалифицированных наставников дети обучаются навыкам инженерных специальностей на современном высокотехнологическом оборудовании. Обучение построено по проектному принципу, на основе практических кейсов: от теории сразу к практике.

## 1.2. Структуру предприятия, функции подразделения и взаимосвязь с другими подразделениями.

Рисунок 1.

Структура предприятия



## 1.3. Список задач подразделения в сфере информационных технологий.

В список задач подразделения нашего подразделения входит создание и поддержка программ для улучшения и поддержки уровня дачи образования ученикам в детский технопарке «Кванториум».

Мы разрабатываем программное обеспечение для «Кванториума», которое помогает преподавателям проводить уроки.

В настоящее время мы разрабатываем и поддерживаем несколько инструментов для повышения качества образования в школах, среди которых:

· Система оценки обучения учеников.

# Содержание выполняемых видов работ.

## 2.1. Анализ существующих в подразделении локальных и распределенных баз данных, приложений баз данных

На данный момент в подразделении отсутствуют базы данных, приложения базы данных.

## 2.2. Получение задания на конкретную разработку

Задание состоит в проектировании и создании программы, которая будет помогать проверять задания студентов.

Необходимо разработать настольное приложение для проверки заданий. Программа будет состоять из интерфейса, который упрощает поиск студентов и заданий. В приложении можно добавлять, удалять и изменять задаваемые параметры в колонке студентов и заданий.

Инструменты для реализации клиентской части: Free Pascal + Lazarus

Инструменты для реализации серверной части: SQLite

## 2.3. Изучение документооборота, подлежащего автоматизации

Документооборот в отделе формирования и реализации дистанционных технологий представляет из себя настольное приложение для проверки заданий, а по окончанию проверки заданий, выводит отчет обо всех студентов в группе.

Рисунок 2.

Документооборот



## 2.4 Построение IDEF-модели

Рисунок 3.

IDEF-модель 1-й уровень

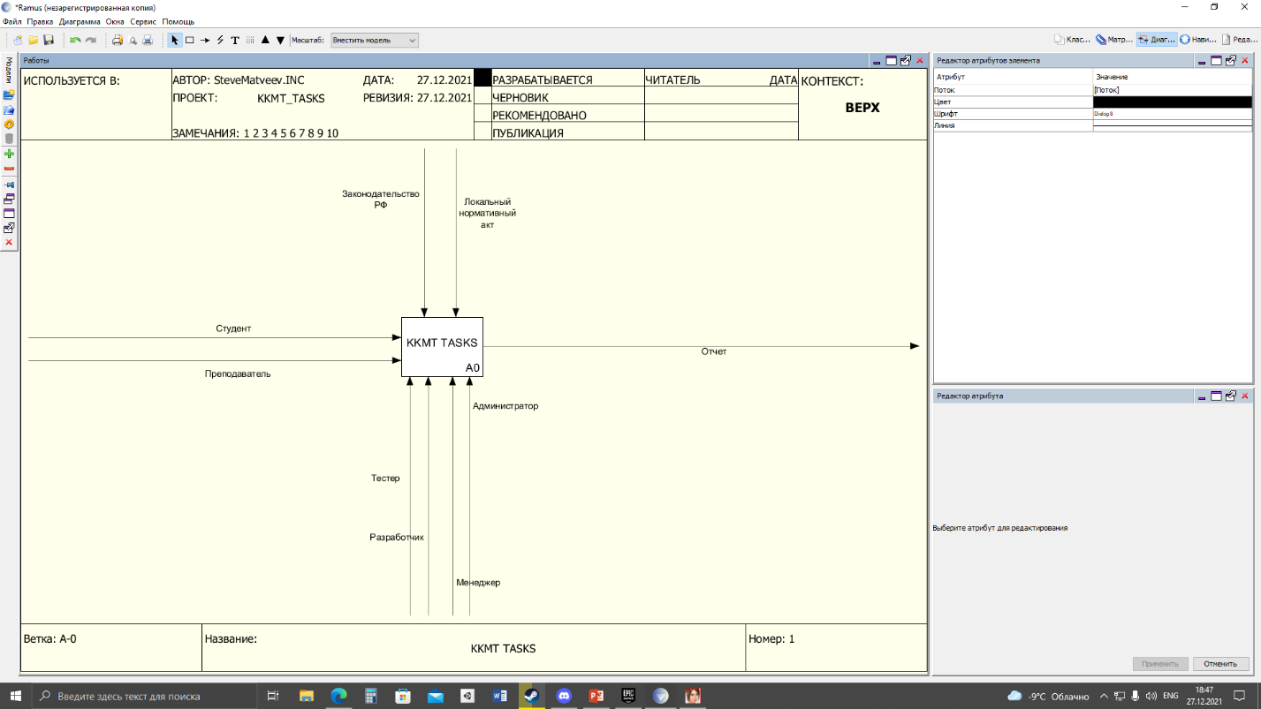
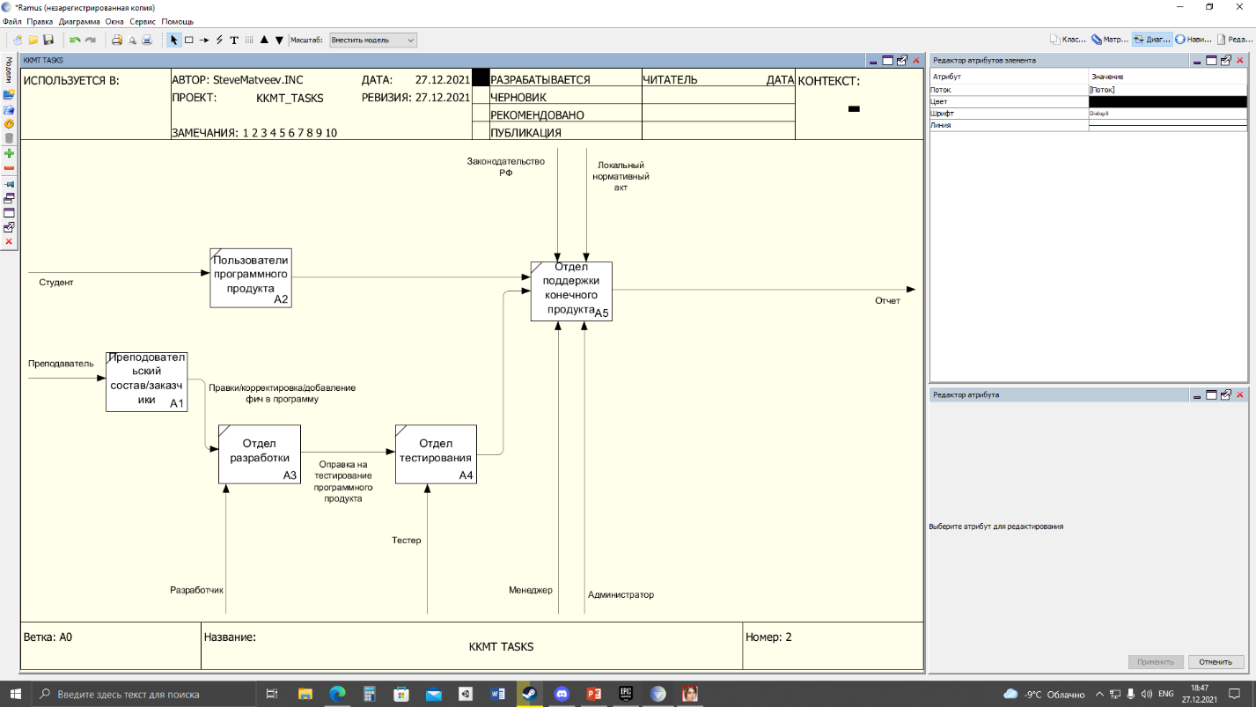


Рисунок 4

IDEF-модель 2-й уровень



* 1. **Составить техническое задание на разработку программного средства.**

### **Наименование программы**

Наименование программы – “KKMT-task 1.0.”.

### **Краткая характеристика области применения**

Необходимо разработать настольное приложение для проверки заданий. Программа будет состоять из интерфейса, который упрощает поиск студентов и заданий. В приложении можно добавлять, удалять и изменять задаваемые параметры в колонке студентов и заданий.

### **Основания для разработки**

Облегчение проверки заданий, преподавателями.

### **Функциональное назначение**

Добавление группы, добавление задания, редактирование и удаление задания, редактирование и удаление группы.

### **Эксплуатационное назначение**

Программа будет эксплуатироваться на устройстве, которое использует ОС Windows 7 и выше. Ограничения к категориям пользователей отсутствуют.

* + 1. **Требования к составу выполняемых функции**

Программа должна содержать 1 окно. Главный экран, который содержит кнопки с верху для работы с фалом. **В разработке.**

### **Требования к организации входных и выходных данных**

Входные данные представляют собой название и содержание текстового файла.

### **Требования к временным характеристикам**

Все операции в программе не должны блокировать графический интерфейс более чем на 1337мс.

### **Требования к обеспечению надежного (устойчивого) функционирования программы**

Программа не должна аварийно завершаться при условии наличия достаточных ресурсов на устройстве (ОЗУ, Темп. ЦП и т.д.)

### **Время восстановления после отказа**

Время восстановления после отказа, вызванным критическим состоянием устройства не должно превышать 5 минут.

### **Отказы из-за некорректных действий оператора**

При некорректных действиях оператора программа не завершиться аварийно.

### **2.5.12. Условия эксплуатации**

Программа должна запускаться как отдельная программа для ОС Windows 7 и выше, где установлен интерпретатор С#.

### **Требования к составу и параметрам технических средств.**

Минимальный состав технических средств:

* ОС: Windows 7 и выше
* ОЗУ: 1 Гб
* ЦП: 1 ГГц.
* ПЗУ: 50 Мб.

### **Технико-экономические показатели**

В разные операционные системы уже как правило встроен текстовый редактор, так что данная программа не будет востребована. Программа сделана для получения опыта в данной сфере.

### **Стадии и этапы разработки**

Разработка состоит из следующих стадий:

* Разработка требований
* Проектирование
* Реализация
* Тестирование
* Внедрение

### **Порядок контроля и приемки**

После разработки всей программы происходит финальное тестирование всего функционала группой контроля качества.

## Разработать модули программного средства в соответствии с ТЗ.

### **2.6.1. Описание главного модуля.**

Главный модуль состоит из главной функции, которая запускается при старте программы инициализирует класс Application и создает главную форму.

Листинг 1 Главная функция

program KkmtTesting;

{$mode objfpc}{$H+}

uses

{$IFDEF UNIX}

cthreads,

{$ENDIF}

{$IFDEF HASAMIGA}

athreads,

{$ENDIF}

Interfaces, // this includes the LCL widgetset

Forms, MainFormUnit, GroupFormUnit, taskformUnit, SettingsFormUnit;

{$R \*.res}

begin

RequireDerivedFormResource:=True;

Application.Scaled:=True;

Application.Initialize;

Application.CreateForm(TMainForm, MainForm);

Application.Run;

end.

### **2.6.2. Описание спецификаций к модулям**

Всего разработано 6 модулей:

* **MainFormUnit**
* **SettingsFormUnit**
* **TaskFormUnit**
* **GroupFormUnit**
* **AppDatabase**
* **Utils**

### **2.6.3. Описание модулей**

**MainFormUnit** – Модуль, который содержит класс главной формы программы.

Рассмотрим функцию запуска обновления выполненных веток и построим блок схему:

Листинг 2. Функция обновления

procedure TMainForm.UpdateStatusButtonClick(Sender: TObject);

begin

if not NowUpdating then

begin

if (SelectedGroup = -1) and (SelectedTask = -1) then exit;

Label2.Caption := StaticText2.Caption;

Label4.Caption := StaticText1.Caption;

Thread := TReportUpdateThread.Create(True, SelectedGroup, SelectedTask);

Thread.Start;

end

else

begin

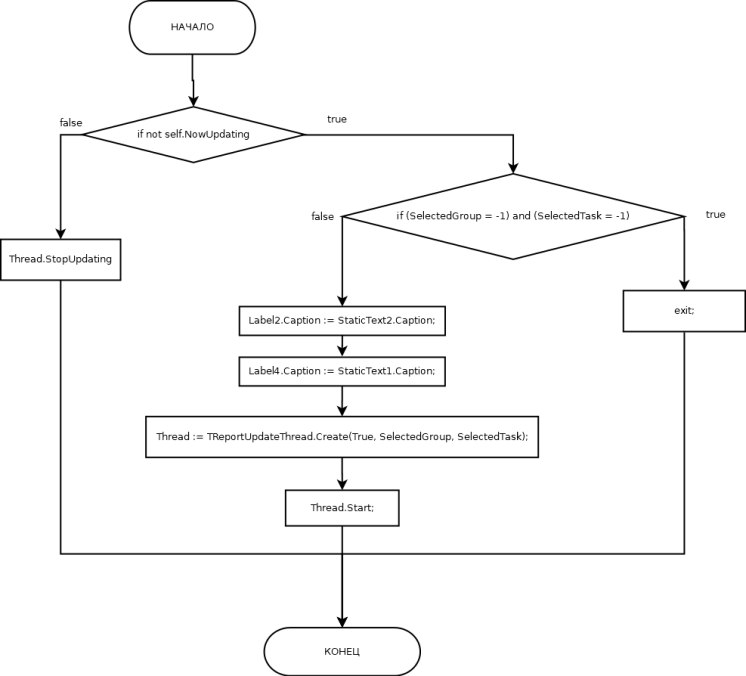
Thread.StopUpdating;

end;

end;

Рисунок 5.

Блок схема класса MainFormUnit



**SettingsFormUnit** - Модуль с классом формы настроек программы.

Имеет два текстовых поля:

* Логин пользователя GitHub
* Персональный токен для этого логина пользователя

При нажатии кнопки "Сохранить" записывает данные в поля в файл config.json в папке программы.

**TaskFormUnit** - Модуль с классом формы для добавления, изменения заданий.

Имеет поле ввода имени задания, а также дерево с ожидаемой структурой ветки начиная от корня ветки.

Также имеет три кнопки:

* Добавить элемент дерева как ребенка выделенного элемента
* Удалить выделенный элемент
* Изменить название выделенного элемента

**GroupFormUnit** - Модуль с классом формы для редактирования, добавления групп и их репозитория для проверки.

Имеет два поля:

* Имя группы
* Уникальный идентификатор репозитория группы

При сохранении данных этой формы выполняется проверка на существование репозитория.

**AppDatabase** - Модуль содержащий классы и функции для работы с базой данных приложения.

Имеет в себе класс TAppDatabase с методами для выполнения запросов с параметрами.

* Exec - для выполнения простого SQL запроса
* ExecOut - для выполнения SQL запроса с возвращением результата

**Utils** - Хранит в себе функции различного назначения.

* CreateGitHubHTTP - для инициализации объекта TFPHttpClient для работы с GitHub API
* OpenDialog - для открытия формы в виде простого диалога

### **2.6.4. Описание тестовых наборов модулей**

В некоторых модулях производится проверка данных на корректность, чтобы сообщить пользователю о проблеме и предотвратить аварийное завершение программы.

В GroupFormUnit происходит проверка на существование репозитория при нажатии на кнопку "Button1".

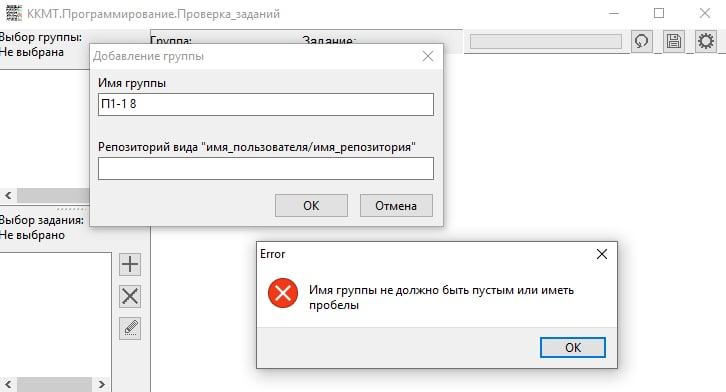
## Разработать пользовательский интерфейс программного средства.

**Тест 1. Проверка имя группы**

Проверка происходит на пустоту имени группы и на пробелы. При непрохождении проверки выводится messagebox с сообщением "Имя группы не должно быть пустым или иметь пробелы".

Рисунок 6.

Ошибка при не правильном вводе группы



**Тест 2. Проверка заполнения вида репозитория**

Проверка на существование репозитория. Если он не существует, то будет показана ошибка "Репозиторий не найден".

Рисунок 7.

Ошибка при неправильном вводе репозитория

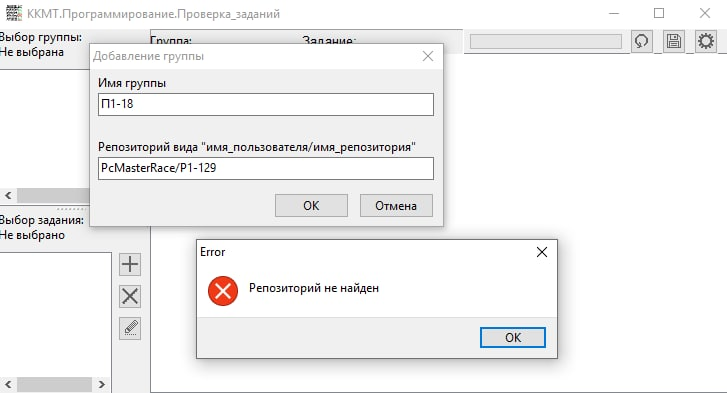


Рисунок 8.

Ошибка при неправильном вводе группы.

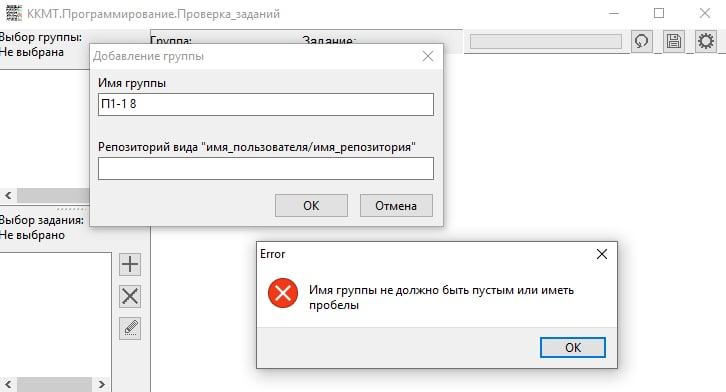


Рисунок 9.

Ошибка при неправильном вводе репозитория.

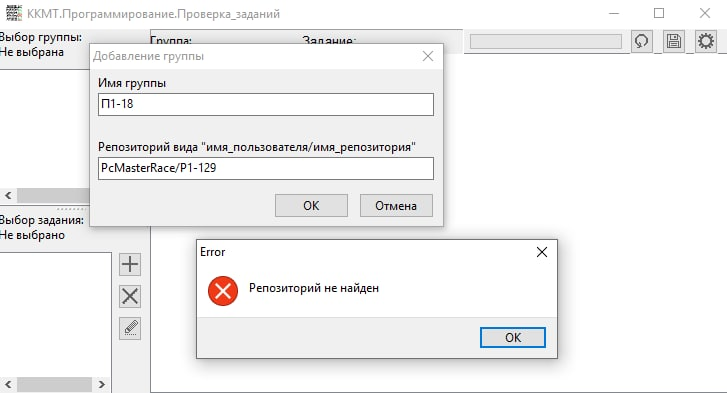


Рисунок 10.

Демонстрация успешного вывода

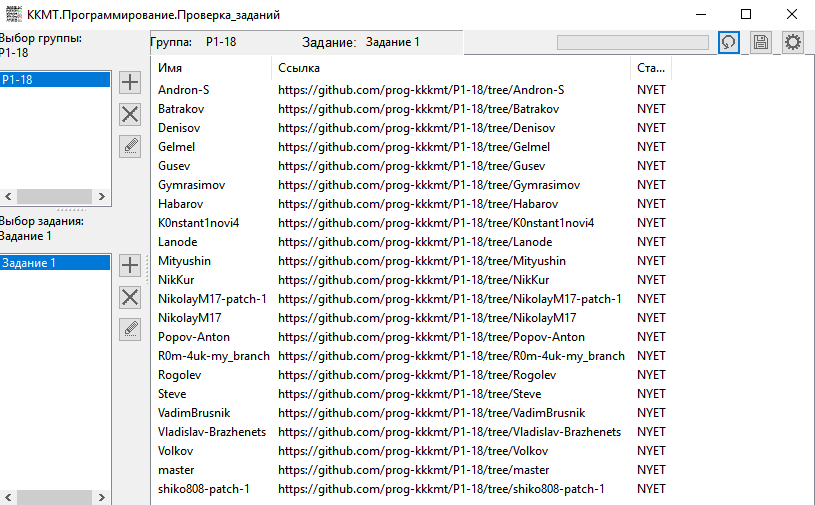
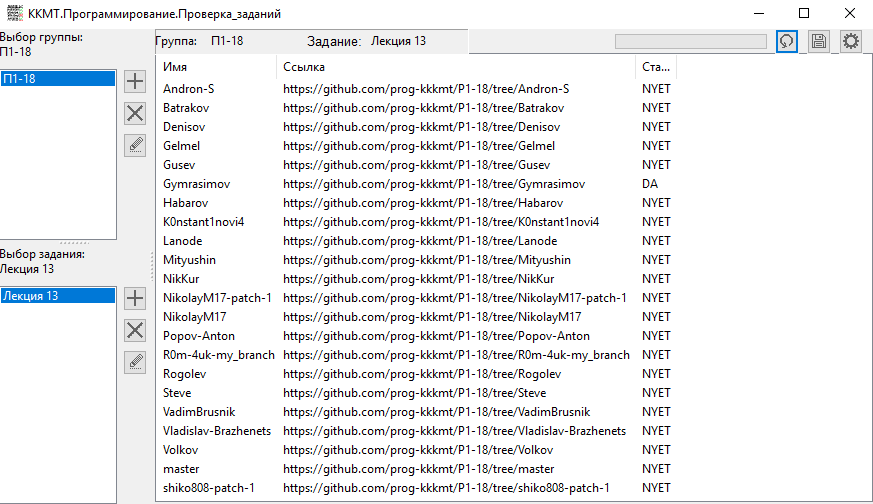


Рисунок 11.

Демонстрация вывода с найденным заданием



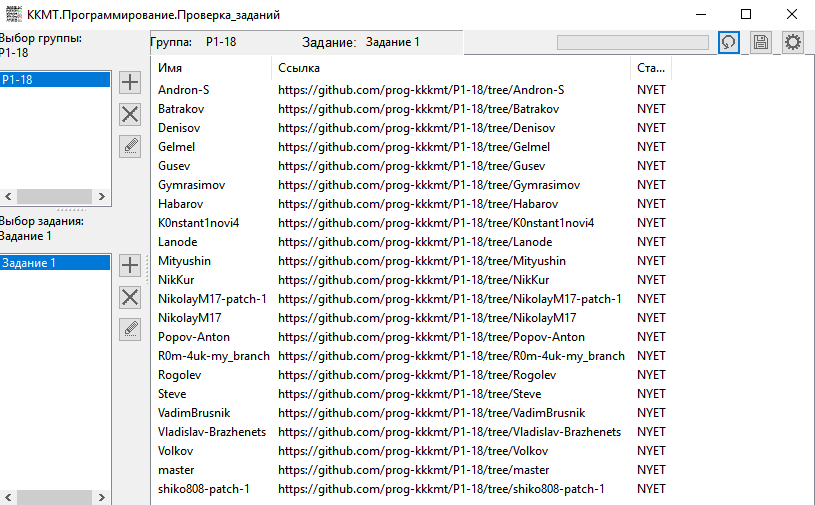
## Провести отладку и тестирование программного средства.

Во время разработки программы мы столкнулись с багом при котором не было проверки репозитория на существование. Поэтому при запуске обновления списка выполнения задания программа выдавала ошибку и не могла продолжить работу дальше.

После того как мы добавили проверку на наличие репозитория в диалог добавления новой группы, баг исчез

Рисунок 12.

Исправление бага



## Разработать комплект технологической документации.

### **Назначение программы**

Данная программа считывает список студентов заданной группы, смотрит есть ли заданное задание на сайте GitHub, и выводит список тех, кто выполнил и не выполнил задание.

### **Условия выполнения программы**

Операционная система: Windows, Linux.

Операционная память: 2 гб.

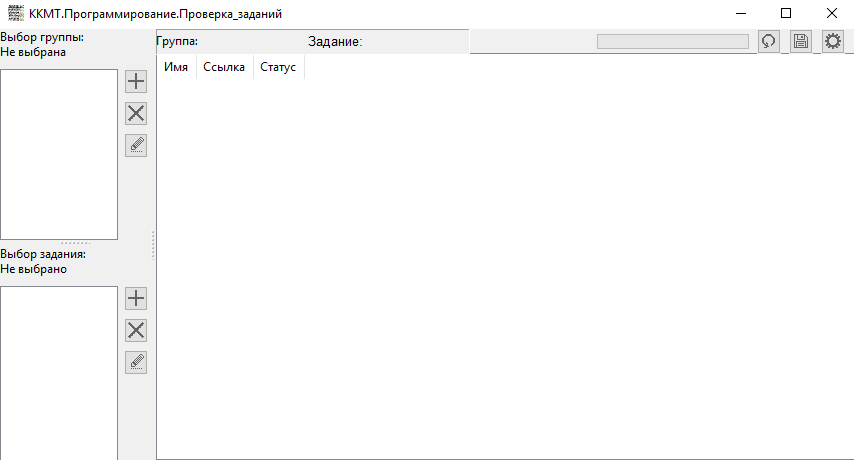
Доступного места на диске: 40 Мб.

### **Выполнение программы**

При запуске программы открывается окно приложения:

Рисунок 13.

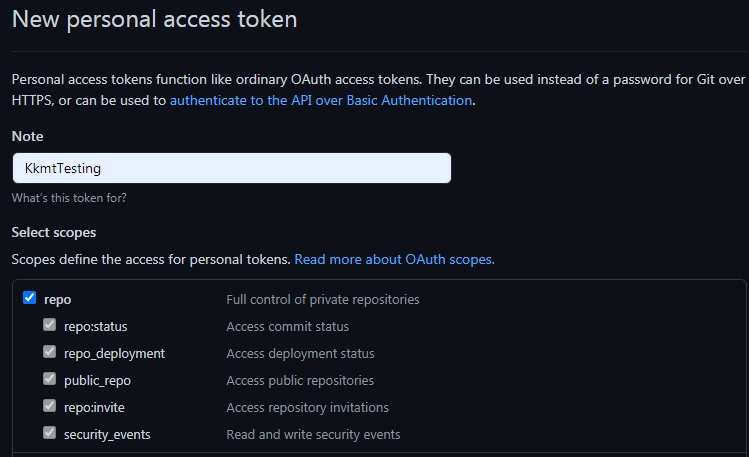
Окно приложения



Для начала, нужно зайти на GitHub для получения token, чтобы авторизоваться в приложении. Получить token можно в GitHub. Чтобы получить токен, нужно нажать на аватарку в правом углу сверху и после нажать Settings. После того, как перешли в свой профиль, нажимаете Developer settings. После перехода в Developer settings, вам нужно нажать Personal access token. Чтобы сгенерировать токен, нажмите Generate new token. После нажатия на кнопку, нужно будет ввести название токена и поставить галочку в кладке repo и после нажать ниже Generate token.

Рисунок 14.

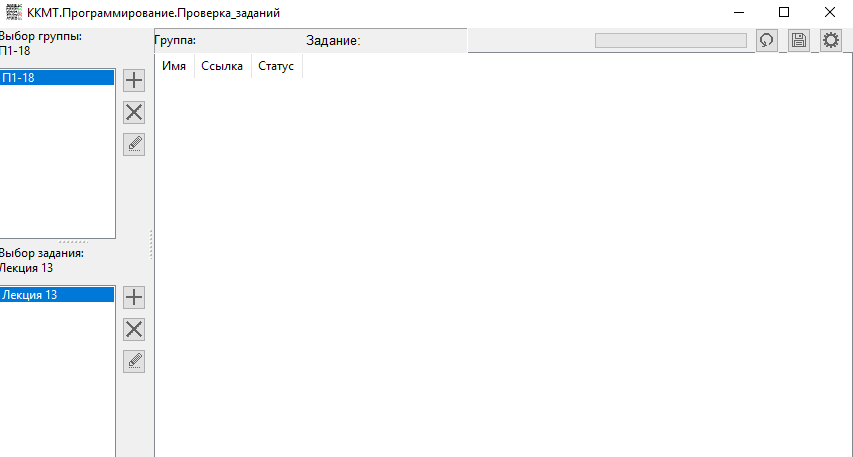
Получение token GitHub API



После того, как введен токен и имя, нужно выбрать группу и задание, для этого надо нажать плюсик.

Рисунок 15.

Выбор группы и задания



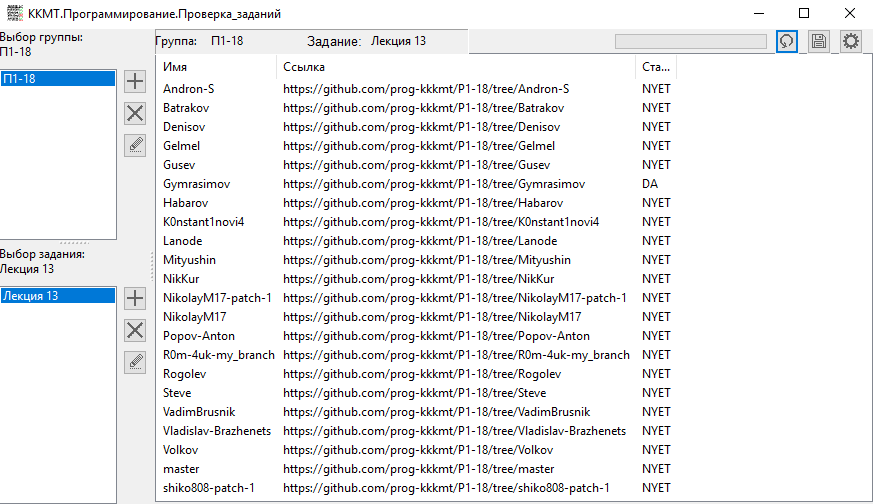
Затем надо ввести имя группы и задание в специальные строки, которые высвечиваются при нажатии плюсиков.

Мы выбрали группу и задание, поэтому чтобы получить результат, вам нужно нажать на закругленную стрелочку.

Статус выдает NYET, когда у студента отсутствует задание, а когда у студента есть это задание, выдает DA.

Рисунок 16.

Вывод данных



Можно сохранить результат, для этого вам нужно нажать на значок дискеты в правом верхнем углу.

Сохраненный результат можно найти по пути: «KKMT.Tasks(текущая версия)\build\(Зависимость от выбора сборки)\reports».

Рисунок 17.

Путь к отчету

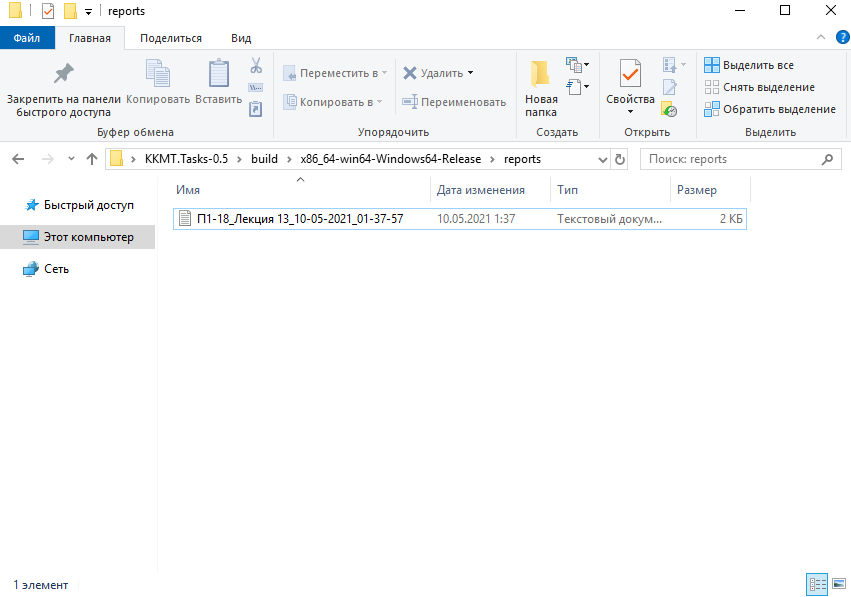


Рисунок 18.

Сохраненный список.

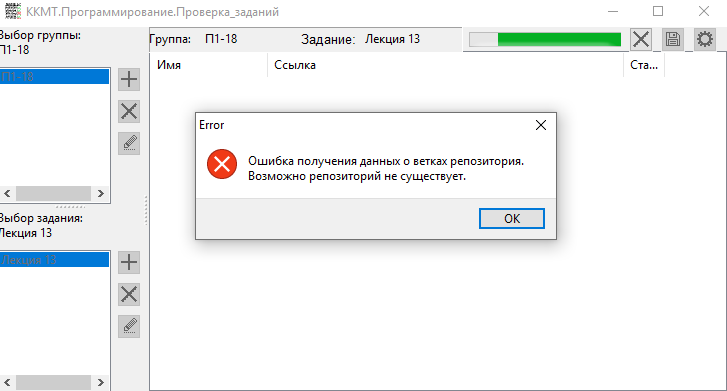


### **Сообщение оператору**

При неправильном вводе данных в форму регистрации или авторизации появиться сообщение с допущенной ошибкой:

Рисунок 19.

Сообщение об ошибке



**Причина:** не введен token GitHub API.

**Действия программы:** программа больше не может обрабатывать результат.

**Действия оператора:** ввести token GitHub API.

1. **Сделать выводы по результатам практики.**

В ходе производственной практики был получен опыт в среде коллективной работы и разработке настольного приложения. Оптимизировали программу, устранив ошибки.

2.2. Структура предприятия, функции подразделения и взаимосвязь с другими подразделениями *(описание подразделений предприятия, можно представить схематично или включить скан структуры предприятия).*

2.3. Список задач подразделения в сфере информационных технологий *(описать деятельность подразделения информационных технологий, например,* *реализация концепции разработки, обеспечение исполнения планов и графиков работ по разработке, внедрение и сопровождение ИТ-сервисов, администрирование, ввод в эксплуатацию и техническая поддержка, совершенствование системы управления предприятием за счет применения современных информационных систем, выявление резервов по повышению эффективности ИТ-сервисов предприятия и т.д.).*

# 3. Проектирование

**3.1. Анализ существующих в подразделении локальных и распределенных баз данных, приложений баз данных (описать используемое на предприятии ПО).**

Не существует.

3.2. Задание на конкретную разработку *(описать будущую разработку).*

3.3. Изучение документооборота, подлежащего автоматизации *(описать применяемый документооборот предприятия, выявить недостатки, достоинства, провести анализ возможностей улучшения работы за счет внедрения Вашего будущего ПС)*.

3.4. Построение IDEF-модели *(описании бизнес-процессов с помощью кейс-средства Ramus, приводим в отчете скрины).*

3.5. Составление технического задания на разработку программного средства.

3.6. Разработка модулей программного средства в соответствии с ТЗ.

3.7. Разработка пользовательского интерфейса программного средства.

3.8. Отладка и тестирование программного средства.

# Заключение

# Дневник производственной практики на предприятии «ККМТ»

# Дневник производственной практики на предприятии

|  |  |  |
| --- | --- | --- |
| Дата | Содержание работы | Подпись руководителя практики от предприятия |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Руководитель: \_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_

Приложение 1. Руководство оператора.

Смотрите приложенный файл с названием «Руководство по использованию программы KkmtTesting.docx»

Приложение 2. Руководство программиста**.**

Приложение 3. Листинг кода**.**

**Листинг 1. MainFormUnit.pas**

unit MainFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, CheckLst,

ExtCtrls, Grids, ValEdit, ComCtrls, SQLite3, jsonparser, fpjson, fphttpclient,

Types, LCLIntf, Buttons,

Utils, AppDatabase,

GroupFormUnit, TaskFormUnit, SettingsFormUnit;

type

{ TMainForm }

// Класс главной формы

TMainForm = class(TForm)

// Кнопка добавления группы

AddGroupButton: TBitBtn;

// Кнопока добавления группы

AddTaskList: TBitBtn;

// Кнопка удаления группы

DeleteGroupButton: TBitBtn;

// Кнопка удаления таска

DeleteTaskButton: TBitBtn;

// Кнопка редактирования группы

EditGroupButton: TBitBtn;

// Группа редактирования таска

EditTaskButton: TBitBtn;

IconList: TImageList;

Panel3: TPanel;

// Кнопка сохранения репорта

SaveStatusReportButton: TBitBtn;

Panel1: TPanel;

Panel2: TPanel;

ProgressBar1: TProgressBar;

// Кнопка настроек

SettingsButton: TBitBtn;

Splitter1: TSplitter;

Splitter2: TSplitter;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

// Лист групп

GroupsList: TListBox;

StaticText1: TStaticText;

StaticText2: TStaticText;

// Лист тасков

TasksList: TListBox;

// Главный лист с статусом выполнения заданий

ListView1: TListView;

TasksPanel: TPanel;

GroupsPanel: TPanel;

// Кнопка обновления репорта

UpdateStatusButton: TBitBtn;

// Нажатие на кнопку добавления группы

procedure AddGroupButtonClick(Sender: TObject);

// Нажатие на кнопку добавления таска

procedure AddTaskListClick(Sender: TObject);

// Нажатие лист групп

procedure GroupsListClick(Sender: TObject);

// Двойное нажатие на лист с статусами заданий

procedure ListView1DblClick(Sender: TObject);

// Нажатие на кнопку сохранения репорта

procedure SaveStatusReportButtonClick(Sender: TObject);

// Нажатие на кнопку настроек

procedure SettingsButtonClick(Sender: TObject);

// Нажатие на лист тасков

procedure TasksListClick(Sender: TObject);

// Нажатие на кнопку обновления статусов заданий

procedure UpdateStatusButtonClick(Sender: TObject);

// Нажатие на кнопку удаления группы

procedure DeleteGroupButtonClick(Sender: TObject);

// Нажатие на кнопку удаления таска

procedure DeleteTaskButtonClick(Sender: TObject);

// Нажатие на кнопку редактирования группы

procedure EditGroupButtonClick(Sender: TObject);

// Нажатие на кнопку редактирования таска

procedure EditTaskButtonClick(Sender: TObject);

// Создание формы

procedure FormCreate(Sender: TObject);

// Функция обновления листа

procedure UpdateList(TableName: String; UpdatingList: TListBox);

//

procedure UpdateCurrentGroup;

//

procedure UpdateCurrentTask;

// Toggle блокировку формы

procedure ChangeFormBlock(AEnabled: Boolean);

private

// Айди выбранной группы

SelectedGroup: Integer;

// Айди выбранного таска

SelectedTask: Integer;

// Айди загруженной группы

LoadedGroup: Integer;

// Айди загруженного таска

LoadedTask: Integer;

public

// Обновляется ли сейчас статус задания

NowUpdating: Boolean;

end;

var

MainForm: TMainForm;

implementation

{$R \*.lfm}

type

{ TReportUpdateThread }

TReportUpdateThread = class(TThread)

private

// Нужна ли остановка обновления

FStop: Boolean;

// Количество веток для проверки

FMax: Integer;

// Уже проверено веток

FProgress: Integer;

// Айди выбранной группы

FSelectedGroup: Integer;

// Айди выбранного таска

FSelectedTask: Integer;

// Лист итемов для добавления в вывод

FListItemToAdd: TListItem;

// Изменить прогресс бар

procedure ChangeProgressBar;

// Обновить прогресс

procedure UpdateProgress;

// Проверка была выполнена успешно

procedure SuccessfulExecute;

protected  
 // Главная функция потока

procedure Execute; override;

// Вызовится в любом случае после завершения работы

procedure AfterExecute(Sender: TObject);

public

constructor Create(CreateSuspended: Boolean; SelectedGroup, SelectedTask: Integer);

// Запуск проверки

procedure Start;

// Остановка проверки

procedure StopUpdating;

end;

var

// Поток проверки заданий

Thread: TReportUpdateThread;

{ TMainForm }

procedure TMainForm.FormCreate(Sender: TObject);

begin

UpdateList('groups', GroupsList);

UpdateList('tasks', TasksList);

SelectedGroup := -1;

SelectedTask := -1;

LoadedGroup := -1;

LoadedTask := -1;

end;

procedure TMainForm.UpdateStatusButtonClick(Sender: TObject);

begin

if not NowUpdating then

begin

if (SelectedGroup = -1) and (SelectedTask = -1) then exit;

Label2.Caption := StaticText2.Caption;

Label4.Caption := StaticText1.Caption;

Thread := TReportUpdateThread.Create(True, SelectedGroup, SelectedTask);

Thread.Start;

end

else

begin

Thread.StopUpdating;

end;

end;

procedure TMainForm.SaveStatusReportButtonClick(Sender: TObject);

var

AppDB: TAppDatabase;

ResultTable: TTable;

GroupName, TaskName: String;

ReportContents: TStringList;

FileName: String;

ListItem: TListItem;

begin

if (LoadedGroup = -1) or (LoadedTask = -1) then

exit;

AppDB := TAppDatabase.Create;

ResultTable := AppDB.ExecOut('SELECT name FROM groups WHERE id=:id', [par('id', LoadedGroup)]);

GroupName := ResultTable[0].KeyData['name'];

ResultTable := AppDB.ExecOut('SELECT name FROM tasks WHERE id=:id', [par('id', LoadedTask)]);

TaskName := ResultTable[0].KeyData['name'];

ResultTable.Free;

AppDB.Free;

ReportContents := TStringList.Create;

FileName := Format('%s\_%s\_%s',

[GroupName, TaskName, FormatDateTime('DD-MM-YYYY\_hh-nn-ss', Now)]);

ReportContents.Add('KKMT TASKS REPORT');

ReportContents.Add(Format('%s', [FileName]));

ReportContents.Add('');

for ListItem in ListView1.Items do

ReportContents.Add(Format('%s , %s , %s',

[ListItem.Caption, ListItem.SubItems[0], ListItem.SubItems[1]]));

ForceDirectories(Application.Location+'/reports');

ReportContents.SaveToFile(Application.Location+'/reports/'+FileName+'.txt');

end;

procedure TMainForm.SettingsButtonClick(Sender: TObject);

var

ModalForm: TSettingsForm;

begin

ModalForm := TSettingsForm.Create(nil);

ModalForm.ShowModal;

ModalForm.Free;

end;

procedure TMainForm.ListView1DblClick(Sender: TObject);

begin

if ListView1.ItemIndex < 0 then

exit;

OpenURL(ListView1.Items[ListView1.ItemIndex].SubItems.Strings[0]);

end;

{ === GroupsPanel === }

procedure TMainForm.GroupsListClick(Sender: TObject);

begin

if GroupsList.ItemIndex < 0 then

exit;

SelectedGroup := TIntObj(GroupsList.Items.Objects[GroupsList.ItemIndex]).I;

StaticText2.Caption := GroupsList.GetSelectedText;

end;

procedure TMainForm.AddGroupButtonClick(Sender: TObject);

var

ModalForm: TGroupForm;

begin

ModalForm := TGroupForm.Create(MainForm);

ModalForm.ShowModal;

ModalForm.Free;

//OpenDialog(TGroupForm);

UpdateList('groups', GroupsList);

end;

procedure TMainForm.DeleteGroupButtonClick(Sender: TObject);

var

AppDB: TAppDatabase;

begin

if GroupsList.ItemIndex >= 0 then

begin

AppDB := TAppDatabase.Create;

AppDB.Exec('DELETE FROM groups WHERE id=:id',

[par('id', TIntObj(GroupsList.Items.Objects[GroupsList.ItemIndex]).I)]);

UpdateCurrentGroup;

UpdateList('groups', GroupsList);

end;

end;

procedure TMainForm.EditGroupButtonClick(Sender: TObject);

var

ModalForm: TGroupForm;

begin

if GroupsList.ItemIndex >= 0 then

begin

ModalForm := TGroupForm.Create(MainForm, TIntObj(GroupsList.Items.Objects[GroupsList.ItemIndex]).I);

ModalForm.ShowModal;

if ModalForm.ModalResult = mrOK then

begin

UpdateCurrentGroup;

UpdateList('groups', GroupsList);

end;

ModalForm.Free;

end;

end;

{ === TasksPanel === }

procedure TMainForm.TasksListClick(Sender: TObject);

begin

if TasksList.ItemIndex < 0 then

exit;

SelectedTask := TIntObj(TasksList.Items.Objects[TasksList.ItemIndex]).I;

StaticText1.Caption := TasksList.GetSelectedText;

end;

procedure TMainForm.AddTaskListClick(Sender: TObject);

var

ModalForm: TTaskForm;

begin

ModalForm := TTaskForm.Create(MainForm);

ModalForm.ShowModal;

ModalForm.Free;

UpdateList('tasks', TasksList);

end;

procedure TMainForm.DeleteTaskButtonClick(Sender: TObject);

var

AppDB: TAppDatabase;

begin

if TasksList.ItemIndex >= 0 then

begin

AppDB := TAppDatabase.Create;

AppDB.Exec('DELETE FROM tasks WHERE id=:id',

[par('id', TIntObj(TasksList.Items.Objects[TasksList.ItemIndex]).I)]);

UpdateCurrentTask;

UpdateList('tasks', TasksList);

end;

end;

procedure TMainForm.EditTaskButtonClick(Sender: TObject);

var

ModalForm: TTaskForm;

begin

if TasksList.ItemIndex >= 0 then

begin

ModalForm := TTaskForm.Create(MainForm, TIntObj(TasksList.Items.Objects[TasksList.ItemIndex]).I);

ModalForm.ShowModal;

if ModalForm.ModalResult = mrOk then

begin

UpdateCurrentTask;

UpdateList('tasks', TasksList);

end;

ModalForm.Free;

end;

end;

procedure TMainForm.UpdateList(TableName: String; UpdatingList: TListBox);

var

AppDB: TAppDatabase;

Table: TTable;

Row: TRow;

begin

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT \* FROM '+TableName+';');

UpdatingList.Clear;

for Row in Table do

begin

UpdatingList.Items.AddObject(Row.KeyData['name'], TIntObj.Create(Row.KeyData['id']));

end;

end;

procedure TMainForm.UpdateCurrentGroup;

var

AppDB: TAppDatabase;

Table: TTable;

begin

if SelectedGroup <> -1 then

begin

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT \* FROM groups WHERE id=:id;', [par('id', SelectedGroup)]);

if Table.Count = 0 then

begin

SelectedGroup := -1;

Label2.Caption := '';

end

else

begin

Label2.Caption := Table.Items[0].KeyData['name'];

end;

end;

if LoadedGroup <> -1 then

begin

Table := AppDB.ExecOut('SELECT \* FROM groups WHERE id=:id;', [par('id', LoadedGroup)]);

if Table.Count = 0 then

begin

LoadedGroup := -1;

ListView1.Clear;

end;

end;

Table.Free;

end;

procedure TMainForm.UpdateCurrentTask;

var

AppDB: TAppDatabase;

Table: TTable;

begin

if SelectedTask <> -1 then

begin

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', SelectedTask)]);

if Table.Count = 0 then

begin

SelectedTask := -1;

Label4.Caption := '';

end

else

begin

Label4.Caption := Table.Items[0].KeyData['name'];

end;

end;

if LoadedTask <> -1 then

begin

Table := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', LoadedTask)]);

if Table.Count = 0 then

begin

LoadedTask := -1;

ListView1.Clear;

end;

end;

Table.Free;

end;

procedure TMainForm.ChangeFormBlock(AEnabled: Boolean);

begin

GroupsList.Enabled := AEnabled;

AddGroupButton.Enabled := AEnabled;

DeleteGroupButton.Enabled := AEnabled;

EditGroupButton.Enabled := AEnabled;

TasksList.Enabled := AEnabled;

AddTaskList.Enabled := AEnabled;

DeleteTaskButton.Enabled := AEnabled;

EditTaskButton.Enabled := AEnabled;

SaveStatusReportButton.Enabled := AEnabled;

SettingsButton.Enabled := AEnabled;

end;

{ TReportUpdateThread }

constructor TReportUpdateThread.Create(CreateSuspended: Boolean; SelectedGroup, SelectedTask: Integer);

begin

inherited Create(CreateSuspended);

FreeOnTerminate := True;

FStop := False;

FProgress := 0;

FSelectedGroup := SelectedGroup;

FSelectedTask := SelectedTask;

FListItemToAdd := nil;

OnTerminate := @AfterExecute;

MainForm.ProgressBar1.Style := TProgressBarStyle.pbstMarquee;

MainForm.ProgressBar1.Position := 0;

MainForm.ListView1.Clear;

end;

procedure TReportUpdateThread.Start;

begin

MainForm.NowUpdating := True;

MainForm.UpdateStatusButton.ImageIndex := 1;

MainForm.ChangeFormBlock(False);

inherited Start;

end;

procedure TReportUpdateThread.StopUpdating;

begin

FStop := True;

end;

procedure TReportUpdateThread.AfterExecute(Sender: TObject);

begin

MainForm.ProgressBar1.Style := TProgressBarStyle.pbstNormal;

MainForm.ProgressBar1.Position := 0;

MainForm.NowUpdating := False;

MainForm.UpdateStatusButton.ImageIndex := 3;

MainForm.ChangeFormBlock(True);

end;

procedure TReportUpdateThread.ChangeProgressBar;

// this method is executed by the mainthread and can therefore access all GUI elements.

begin

MainForm.ProgressBar1.Style := TProgressBarStyle.pbstNormal;

MainForm.ProgressBar1.Max := FMax;

end;

procedure TReportUpdateThread.UpdateProgress;

// this method is executed by the mainthread and can therefore access all GUI elements.

begin

MainForm.ProgressBar1.StepIt;

MainForm.ListView1.Items.AddItem(FListItemToAdd);

end;

procedure TReportUpdateThread.SuccessfulExecute;

begin

MainForm.LoadedGroup := FSelectedGroup;

MainForm.LoadedTask := FSelectedTask;

end;

procedure TReportUpdateThread.Execute;

var

HTTP: TFPHttpClient;

AppDB: TAppDatabase;

Table: TTable;

GitHubId: String;

HttpResult: String;

BranchesJSON: TJSONArray;

Enum, Enum2: TJSONEnum;

Files: TStringList;

Counter: Integer;

begin

HTTP := CreateGitHubHTTP;

AppDB := TAppDatabase.Create;

Table := AppDB.ExecOut('SELECT github\_id FROM groups WHERE id=:id;', [par('id', FSelectedGroup)]);

if Table.Count <= 0 then

exit;

GitHubId := Table[0].KeyData['github\_id'];

Table := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', FSelectedTask)]);

if Table.Count <= 0 then

exit;

Files := TStringList.Create;

for Enum in TJSONArray(GetJSON(Table[0].KeyData['files'])) do

Files.Add(Enum.Value.AsString);

try

HttpResult := HTTP.Get(Format('https://api.github.com/repos/%s/branches', [GitHubId] ));

except

MessageDlg('Ошибка получения данных о ветках репозитория. Возможно репозиторий не существует.', mtError,

[mbOK], 0);

exit;

end;

BranchesJSON := TJSONArray(GetJSON(HttpResult));

FMax := BranchesJSON.Count;

Synchronize(@ChangeProgressBar);

for Enum in BranchesJSON do

begin

if FStop then break;

FListItemToAdd := TListItem.Create(MainForm.ListView1.Items);

FListItemToAdd.Caption := TJSONObject(enum.Value)['name'].AsString;

FListItemToAdd.SubItems.Add(Format('https://github.com/%s/tree/%s',

[GitHubId, TJSONObject(enum.Value)['name'].AsString]));

HttpResult := HTTP.Get(Format('https://api.github.com/repos/%s/git/trees/%s?recursive=1',

[GitHubId, TJSONObject(enum.Value).GetPath('commit.sha').AsString]));

Counter := Files.Count;

for Enum2 in TJSONObject(GetJSON(HttpResult)).Arrays['tree'] do

begin

if Files.IndexOf(TJSONObject(Enum2.Value).Strings['path']) <> -1 then

Dec(Counter);

end;

if Counter = 0 then

// something good

FListItemToAdd.SubItems.Add('DA')

else

// somethinf bad

FListItemToAdd.SubItems.Add('NYET');

Synchronize(@UpdateProgress);

end;

BranchesJSON.Free;

AppDB.Free;

HTTP.Free;

Table.Free;

Files.Free;

Synchronize(@SuccessfulExecute);

end;

end.

**Листинг 2. GroupFormUnit.pas**

unit GroupFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,

fphttpclient,

AppDatabase, Utils;

type

{ TGroupForm }

TGroupForm = class(TForm)

// OK кнопка

Button1: TButton;

// Кнопка отмены

Button2: TButton;

// Поле имени группы

LabeledEdit1: TLabeledEdit;

// Поле айди репозитория группы

LabeledEdit2: TLabeledEdit;

// Нажатие на ОК кнопку

procedure Button1Click(Sender: TObject);

// Нажатие на кнопку отмены

procedure Button2Click(Sender: TObject);

private

// Айди группы для режима реадктирования

EditId: Integer;

public

constructor Create(TheOwner: TComponent);

constructor Create(TheOwner: TComponent; GroupForEditing: Integer); overload;

// Изменить дизайн формы для редактирования группы

procedure MakeForEdit(GroupId: Integer);

end;

implementation

{$R \*.lfm}

{ TGroupForm }

procedure TGroupForm.Button1Click(Sender: TObject);

var

HTTP: TFPHttpClient;

HttpResult: String;

AppDB: TAppDatabase;

begin

HTTP := CreateGitHubHTTP;

AppDB := TAppDatabase.Create;

//ShowMessage(Format('https://api.github.com/repos/%s', [LabeledEdit2.Text]));

if String(LabeledEdit1.Text).IsEmpty or String(LabeledEdit1.Text).Contains(' ') then

begin

MessageDlg('Имя группы не должно быть пустым или иметь пробелы', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

try

HttpResult := HTTP.Get(Format('https://api.github.com/repos/%s', [LabeledEdit2.Text]));

except

if HTTP.ResponseStatusCode = 404 then

begin

MessageDlg('Репозиторий не найден', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end

else

begin

MessageDlg('Ошибка получения данных о репозитории. Возможно репозиторий не существует.', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

end;

try

if EditId = -1 then

AppDB.Exec('INSERT INTO groups (name, github\_id) VALUES (:name, :rep\_id);',

[par('name', LabeledEdit1.Text), par('rep\_id', LabeledEdit2.Text)])

else

AppDB.Exec('UPDATE groups SET name=:name, github\_id=:rep\_id WHERE id=:id;',

[par('name', LabeledEdit1.Text), par('rep\_id', LabeledEdit2.Text), par('id', EditId)] );

except

on E: Exception do

begin

MessageDlg('Ошибка при записи в базу данных.'+LineEnding+E.Message, mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

end;

end;

procedure TGroupForm.Button2Click(Sender: TObject);

begin

end;

constructor TGroupForm.Create(TheOwner: TComponent);

begin

inherited Create(TheOwner);

EditId := -1;

end;

constructor TGroupForm.Create(TheOwner: TComponent; GroupForEditing: Integer);

begin

inherited Create(TheOwner);

MakeForEdit(GroupForEditing);

end;

procedure TGroupForm.MakeForEdit(GroupId: Integer);

var

AppDB: TAppDatabase;

ResultTable: TTable;

begin

AppDB := TAppDatabase.Create;

ResultTable := AppDB.ExecOut('SELECT \* FROM groups WHERE id=:id;', [par('id', GroupId)]);

Self.LabeledEdit1.Text := ResultTable[0].KeyData['name'];

Self.LabeledEdit2.Text := ResultTable[0].KeyData['github\_id'];

Self.Caption := 'Изменение группы';

Self.EditId := GroupId;

end;

end.

**Листинг 3. TaskFormUnit.pas**

unit TaskFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,

CheckLst, ColorBox, ShellCtrls, ComCtrls, Buttons, AppDatabase, fpjson;

type

{ TTaskForm }

TTaskForm = class(TForm)

// ОК кнопка

Button1: TButton;

// Кнопка отмены

Button2: TButton;

// Кнопка добавления дочерней ноды имени файла

Button3: TBitBtn;

// Кнопка удаления ноды имени файла

Button4: TBitBtn;

Label1: TLabel;

// Имя таска

LabeledEdit1: TLabeledEdit;

Panel1: TPanel;

// Дерево файлов

TreeView1: TTreeView;

// Нажатие на кнопку сохранения

procedure Button1Click(Sender: TObject);

// Нажатие на кнопку отмены

procedure Button2Click(Sender: TObject);

// Нажатие на кнопку добавления ноды

procedure Button3Click(Sender: TObject);

// Нажатие на кнопку удаления ноды

procedure Button4Click(Sender: TObject);

// Выделение ноды

procedure TreeView1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

private

// Айди таска для режима редактирования

EditId: Integer;

// Создание массива путей для каждого файла

function TreeToPathsGenerator(Tree: TTreeNodes): TStringList;

// Создание дерева из массива путей

function TreeFromPathsGenerator(Paths: TStringList): TTreeNodes;

public

constructor Create(TheOwner: TComponent);

constructor Create(TheOwner: TComponent; TaskForEditing: Integer); overload;

// Изменить дизайн формы для режима редактирования

procedure MakeForEdit(TaskId: Integer);

end;

implementation

{$R \*.lfm}

{ TTaskForm }

procedure TTaskForm.Button3Click(Sender: TObject);

var

Filename: String;

begin

Filename := InputBox('Введите имя файла',

'Введите имя файла для проверки', '');

if not Filename.IsEmpty then

TreeView1.Items.AddChild(TreeView1.Selected, Filename);

end;

procedure TTaskForm.Button4Click(Sender: TObject);

begin

if TreeView1.Selected <> nil then

TreeView1.Items.Delete(TreeView1.Selected);

end;

procedure TTaskForm.TreeView1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

var

i: Integer;

begin

if not Assigned(TreeView1.GetNodeAt(X, Y)) then

for i := 0 to TreeView1.Items.Count - 1 do

TreeView1.Items[i].Selected := false;

end;

procedure TTaskForm.Button1Click(Sender: TObject);

var

AppDB: TAppDatabase;

FilePath: String;

json: TJSONArray;

begin

if Length(LabeledEdit1.Text) <= 0 then

begin

MessageDlg('Введите имя задания', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

if TreeView1.Items.Count <= 0 then

begin

MessageDlg('Задайте структуру задания', mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

json := TJSONArray.Create;

AppDB := TAppDatabase.Create;

for FilePath in TreeToPathsGenerator(TreeView1.Items) do

json.Add(FilePath.Substring(1));

//ShowMessage(json.AsJSON);

try

if EditId = -1 then

AppDB.Exec('INSERT INTO tasks (name, files) VALUES (:name, :files);', [par('name', LabeledEdit1.Text), par('files', json.AsJSON)] )

else

AppDB.Exec('UPDATE tasks SET name=:name, files=:files WHERE id=:id;',

[par('name', LabeledEdit1.Text), par('files', json.AsJSON), par('id', EditId)] );

except

on E: Exception do

begin

MessageDlg('Ошибка при записи в базу данных.'+LineEnding+E.Message, mtError,

[mbOK], 0);

ModalResult := mrNone;

exit;

end;

end;

end;

procedure TTaskForm.Button2Click(Sender: TObject);

begin

end;

function TTaskForm.TreeToPathsGenerator(Tree: TTreeNodes): TStringList;

var Paths: TStringList;

procedure TreeWalker(Node: TTreeNode; PathString: String);

var

i: Integer;

begin

PathString += '/'+Node.Text;

if Node.HasChildren then

for i:=0 to Node.Count-1 do

TreeWalker(Node.Items[i], PathString)

else

Paths.Add(PathString);

end;

var

Node: TTreeNode;

begin

Paths := TStringList.Create;

for Node in Tree do

begin

if Node.Parent = nil then

TreeWalker(Node, '');

end;

Result := Paths;

end;

function TTaskForm.TreeFromPathsGenerator(Paths: TStringList): TTreeNodes;

var

Path: String;

PathParsed: TStringArray;

i, j: Integer;

Node, ChildNode: TTreeNode;

begin

Result := TreeView1.Items;

for Path in Paths do

begin

PathParsed := Path.Split(['/'], TStringSplitOptions.ExcludeEmpty);

Node := Result.FindNodeWithText(PathParsed[0]);

if (Node = nil) or (Node.Parent <> nil) then

Node := Result.AddChild(nil, PathParsed[0]);

for i := 1 to Length(PathParsed)-1 do

begin

ChildNode := Node.FindNode(PathParsed[i]);

if ChildNode = nil then

ChildNode := Result.AddChild(Node, PathParsed[i]);

Node := ChildNode;

end;

end;

end;

constructor TTaskForm.Create(TheOwner: TComponent);

begin

inherited Create(TheOwner);

EditId := -1;

end;

constructor TTaskForm.Create(TheOwner: TComponent; TaskForEditing: Integer);

begin

inherited Create(TheOwner);

MakeForEdit(TaskForEditing);

end;

procedure TTaskForm.MakeForEdit(TaskId: Integer);

var

AppDB: TAppDatabase;

ResultTable: TTable;

Paths: TStringList;

Enum: TJSONEnum;

begin

AppDB := TAppDatabase.Create;

ResultTable := AppDB.ExecOut('SELECT \* FROM tasks WHERE id=:id;', [par('id', TaskId)]);

Self.LabeledEdit1.Text := ResultTable[0].KeyData['name'];

Paths := TStringList.Create;

for Enum in TJSONArray(GetJSON(ResultTable[0].KeyData['files'])) do

Paths.Add(Enum.Value.AsString);

Self.TreeFromPathsGenerator(Paths);

Self.Caption := 'Изменение задания';

Self.EditId := TaskId;

end;

end.

**Листинг 4. AppDatabase.pas**

unit AppDatabase;

interface

uses

fpjson, sqldb, db, SQLite3Conn, sysutils, Generics.Collections, fgl, Variants, SQLite3;

type

// SQL параметр вы виде пары имя-значение

TParamPair = specialize TPair<String, Variant, String>;

// Динамический массив параметров

TParamsArray = Array of TParamPair;

// Строка таблицы

TRow = specialize TFPGMap<String, Variant>;

// Таблица

TTable = specialize TList<TRow>;

// Класс базы данных приложения

TAppDatabase = class (TInterfacedObject)

// Объект соединения с базой данных

Connection: TSQLite3Connection;

// Объект транзакции

Transaction: TSQLTransaction;

// Объект запроса

Query: TSQLQuery;

// Выполнить SQL запрос с параметрами без возврата результата

procedure Exec(AQuery: String; AParams: TParamsArray);

// Выполнить SQL запрос без параметров и без возврата результата

procedure Exec(AQuery: String);

// Выполнить SQL запрос с параметрами и с возвратом результата

function ExecOut(AQuery: String; AParams: TParamsArray): TTable;

// Выполнить SQL запрос без параметров с возвратом результата

function ExecOut(AQuery: String): TTable;

constructor Create;

destructor Free;

end;

// Функция для упрощенного создания объекта TParamPair

function par(AName: String; AValue: Variant): TParamPair;

implementation

// ===== TDatabase =====

procedure TAppDatabase.Exec(AQuery: String; AParams: TParamsArray);

var

Param: TParamPair;

begin

Transaction.StartTransaction;

Query.SQL.Text := AQuery;

if Length(AParams) > 0 then

for Param in AParams do

Query.ParamByName(Param.Key).Value := Param.Value;

Query.Prepare;

try

Query.ExecSQL;

finally

Transaction.Commit;

Query.Clear;

Transaction.EndTransaction;

end;

end;

procedure TAppDatabase.Exec(AQuery: String);

begin

Exec(AQuery, nil);

end;

function TAppDatabase.ExecOut(AQuery: String; AParams: TParamsArray): TTable;

var

Param: TParamPair;

Row: TRow;

Field: TField;

begin

Transaction.StartTransaction;

Query.SQL.Text := AQuery;

if Length(AParams) > 0 then

for Param in AParams do

Query.ParamByName(Param.Key).Value := Param.Value;

Query.Prepare;

try

Query.Open;

Result := TTable.Create;

while not Query.EOF do

begin

Row := TRow.Create;

for Field in Query.Fields do

Row.Add(Field.FieldName, Field.Value);

Result.Add(Row);

Query.Next;

end;

finally

Query.Close;

Transaction.Commit;

Query.Clear;

Transaction.EndTransaction;

end;

end;

function TAppDatabase.ExecOut(AQuery: String): TTable;

begin

Result := ExecOut(AQuery, nil);

end;

constructor TAppDatabase.Create;

begin

//if not FileExists('./application.db') then

//begin

// FileCreate('./application.db');

//end;

Connection := TSQLite3Connection.Create(nil);

Transaction := TSQLTransaction.Create(nil);

Query := TSQLQuery.Create(nil);

Connection.DatabaseName := './application.db';

Connection.Transaction := Transaction;

Query.DataBase := Connection;

Connection.Open;

Exec('CREATE TABLE IF NOT EXISTS "groups" ("name" TEXT NOT NULL UNIQUE,"github\_id" TEXT NOT NULL UNIQUE,"id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE)');

Exec('CREATE TABLE IF NOT EXISTS "tasks" ("id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,"name" TEXT NOT NULL,"files" TEXT NOT NULL)');

end;

destructor TAppDatabase.Free;

begin

Transaction.EndTransaction;

Connection.Connected := False;

Connection.Free;

Transaction.Free;

Query.Free;

end;

function par(AName: String; AValue: Variant): TParamPair;

begin

Result.Key := AName;

Result.Value := AValue;

end;

end.

**Листинг 5. SettingFormUnit.pas**

unit SettingsFormUnit;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls;

type

{ TSettingsForm }

TSettingsForm = class(TForm)

// Кнопка сохранения

Button1: TButton;

// Кнопка отмены

Button2: TButton;

// Поле с именем пользователя GitHub

LabeledEdit1: TLabeledEdit;

// Поле с токеном пользователя GitHub

LabeledEdit2: TLabeledEdit;

// Нажатие на кнопку сохранения

procedure Button1Click(Sender: TObject);

// Нажатие на кнопку отмены

procedure Button2Click(Sender: TObject);

// Создания формы

procedure FormCreate(Sender: TObject);

private

// Сохранение настроек в конфиг

procedure LoadConfig;

// Загрузка настроек из конфига

procedure SaveConfig;

public

end;

implementation

{$R \*.lfm}

uses

jsonConf;

{ TSettingsForm }

procedure TSettingsForm.Button2Click(Sender: TObject);

begin

Self.Close;

end;

procedure TSettingsForm.FormCreate(Sender: TObject);

begin

LoadConfig;

end;

procedure TSettingsForm.Button1Click(Sender: TObject);

begin

SaveConfig;

Self.Close;

end;

procedure TSettingsForm.LoadConfig;

var

c: TJSONConfig;

begin

c := TJSONConfig.Create(nil);

c.Filename := Application.Location+'/config.json';

LabeledEdit1.Text := c.GetValue('/github/login', '');

LabeledEdit2.Text := c.GetValue('/github/token', '');

c.Free;

end;

procedure TSettingsForm.SaveConfig;

var

c: TJSONConfig;

begin

c := TJSONConfig.Create(nil);

c.Filename := Application.Location+'/config.json';

c.SetValue('/github/login', LabeledEdit1.Text);

c.SetValue('/github/token', LabeledEdit2.Text);

c.Free;

end;

end.

**Листинг 6. Utils.pas**

unit Utils;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, Dialogs, fpjson, fphttpclient, Forms;

type

// Класс инкапсуляции объекта Integer

TIntObj = class

private

FI: Integer;

public

property I: Integer Read FI;

constructor Create(IValue: Integer);

end;

// Создание и подготовка HttpClient для работы с GitHub API

function CreateGitHubHTTP: TFPHttpClient;

// Функция быстрого открытия модального окна

procedure OpenDialog(FormClass: TFormClass);

implementation

uses

opensslsockets, jsonConf;

{ TIntObj }

constructor TIntObj.Create(IValue: Integer);

begin

Inherited Create;

FI := IValue;

end;

function CreateGitHubHTTP: TFPHttpClient;

var

c: TJSONConfig;

begin

c := TJSONConfig.Create(nil);

Result := TFPHTTPClient.Create(nil);

c.Filename := Application.Location+'/config.json';

Result.AllowRedirect := True;

Result.AddHeader('User-Agent', 'Mozilla/5.0 (compatible; fpweb)');

Result.AddHeader('Accept', 'application/vnd.github.v3+json');

Result.UserName := c.GetValue('/github/login', '');

Result.Password := c.GetValue('/github/token', '');

end;

procedure OpenDialog(FormClass: TFormClass);

var

ModalForm: TForm;

begin

ModalForm := FormClass.Create(nil);

ModalForm.ShowModal;

ModalForm.Free;

end;

end.